Pergamon

Neural Networks, Vol. 9, No. 1, pp. 83–97, 1996

Printed in Great Britain
0893-6080/96 $15.00+.00

0893-6080(95)00096-8

CONTRIBUTED ARTICLE

Training MLPs Layer by Layer Using an Objective Function for Internal Representations

RÉGIS LENGELLÉ AND THIERRY DENŒUX

Université de Technologie de Compiègne and Lyonnaise de Eaux/Laboratoire d'Informatique Avancée de Compiègne

(*Received* 26 *January* 1993; *accepted* 21 *June* 1995)

Abstract—*A new constructive algorithm for designing and training multilayer perceptrons is proposed. This algorithm involves the optimization of an objective function for internal representations, which does not require any computation of the network's outputs. Coupled with a strategy for recruiting units during the learning process, this concept provides a scheme for training a multilayer network layer by layer, until self-encoding of the pattern categories is achieved in the final, highest-level representations. Two objective functions are proposed. For discrimination problems, recent experimental and theoretical results concerning back-propagation training of networks with one hidden layer and linear outputs suggest the introduction of a particular measure of class separability. For problems involving the approximation of a continuous function, we show that the minimization of the mean squared output error can be achieved by maximizing a statistical measure (the sample coefficient of multiple determination) in the last hidden layer. Simulations are used to illustrate the process of network construction, and to demonstrate the improvements brought by this approach over back-propagation in terms of performance and robustness.*

Keywords—Multilayer perceptrons, Learning algorithm, Internal representation, Architecture, Constructive algorithm, Classification, Approximation, Discriminant analysis.

1. INTRODUCTION

The back-propagation algorithm owes much of its success in challenging real-world applications to its ability to form complex internal representations (Rumelhart et al., 1986). Whereas simple two-layer associative networks can only map similar inputs to similar outputs, the recoding of input patterns in a layer of non-linear hidden units considerably extends the class of implementable mappings. Some time after the introduction of back-propagation as a powerful learning procedure (Rumelhart et al., 1986), this argument received its most striking confirmation from a series of theorems showing that networks with only one layer of non-linear units could approximate virtually any desired mapping with arbitrary accuracy (see, for example, Hornik, 1991).

In the original back-propagation algorithm, the number of hidden units and the connectivity are predetermined, and internal representations are constructed automatically by a gradient descent procedure which attempts to minimize the squares of the differences between the actual and the desired output values summed over the output units and all pairs of input/output vectors (Rumelhart et al., 1986). Although it represents a considerable improvement over previous models in which the hidden layer weights could not be adapted at all (Rosenblatt, 1958), this approach leaves open the question of the optimal dimensionality of the representation space onto which the input patterns are projected as a result of the learning process. The design of a suitable architecture for a neural network in view of performing a given task has been found to be both critical with regard to generalization ability (Hinton, 1990), and extremely difficult to handle more efficiently than through a time-consuming process of trial and error.

In this paper, a novel approach to this problem is proposed. This approach is based on direct optimiza-

Acknowledgements: This work has been supported by EEC funded Esprit II project nr. 5433 (NEUFODI); partners: BIKIT, ARIAI, Elorduy Sancho y Cia, LABEIN, Lyonnaise des Eaux Dumez. The authors wish to express their thanks to the Foundation for Applied Neuroscience Research in Psychiatry (CHS de Rouffach, 68250 Rouffach, France) for providing the sleep analysis data.

Requests for reprints should be sent to T. Denœux, Université, de Technologie de Compiègne, U.R.A. CNRS 817 Heudiasyc, BP 649 F-60206 Compiègne cedex, France; E-mail: tdenoeux@hds.univ-compiegne.fr

tion of an *objective function* for internal representations, which can be computed given a set of examples without specifying the network's outputs. Coupled with a strategy for recruiting units during the learning process, this concept provides a scheme for training a multilayer network layer by layer, until a simple correspondence can be found between the final, highest-level representations and a simple target coding scheme. Two objective functions are proposed. For classification problems, recent experimental and theoretical results (Webb & Lowe, 1990) concerning back-propagation training of networks with one hidden layer and linear outputs suggest the introduction of a measure of class separability commonly used in statistical pattern recognition. For problems involving the approximation of a continuous function, we show that the minimization of the mean squared output error can be achieved by maximizing the sample coefficient of multiple determination in the last hidden layer, which can be done without explicit calculation of the hidden-to-output weights.

The remainder of this paper is organized as follows. In Section 2, some previous approaches to the design of minimal size multilayer networks are briefly reviewed. Section 3 explains why back-propagation networks trained on classification tasks can be viewed as performing non-linear discriminant analysis, which justifies the introduction of class separability as an objective function for internal representations. Our constructive algorithm is then presented in detail in Section 4, after which an extension to the continuous case is proposed (Section 5). Finally, simulation results are presented and discussed in Section 6.

2. DESIGNING THE ARCHITECTURE OF MULTILAYER PERCEPTRONS

As mentioned in the introduction, we know from the so-called "approximation theorems" (Hornik, 1991) that neural networks with sufficiently many units in one hidden layer are in principle capable of performing any approximation task. Although these results have had a tremendous impact from a theoretical point of view, they have not been of much help to the practitioners who want to know which architecture (number of layers, number of units in each layer) is needed for a given application. An important step in this direction has been made by research work which has provided *upper bounds* of the number of threshold or sigmoidal hidden units needed to either dichotomize, or approximately interpolate *any* set of n data points (Sontag, 1991). However, these bounds, resulting from a worst case analysis, lead to considerable overestimation when applied to most of the tasks encountered in practice.

The size determination problem appears to be even more complicated when neural net training is considered from the point of view of *inductive inference*. In that case, the focus is not on how well the network has been able to memorize a set of input–output patterns, but on the extent to which it will be able to *generalize* to unseen cases. In that framework, the function implemented by a network trained on a particular data set can be viewed as a *hypothesis* which accounts for this data. Since a large number of such hypotheses usually exist, there is a need for an *inductive bias* (Haussler, 1988) whereby some hypotheses are preferred a priori over others. Following the assumption that the simplest explanation is the most plausible, the inductive bias is often related to some measure of complexity. One statistical approach to this problem has been provided by Vapnik's framework (Vapnik, 1983), which has been used to propose upper bounds of the probability that a network trained on a number of samples will perform poorly on unseen samples drawn from the same distribution (Baum & Haussler, 1989; Burton & Farris, 1991). These results suggest in particular that, for classification problems, the number of samples needed to achieve a generalization error rate of ϵ is roughly the number of weights divided by ϵ (Baum & Haussler, 1989). These considerations, which are also supported by empirical evidence indicating that small networks are less prone to overfitting than large networks, justify the search for the *minimal size* network capable of performing a given task.

Having recognized the importance of size determination, we are faced with the problem of building networks which are at least close to the minimal one. One strategy for solving this problem is the *constructive* approach in which a small initial network is gradually expanded until the task is considered to be solved. In Mézard and Nadal (1989), Marchand et al. (1990) and Zollner et al. (1992), such schemes are described in the particular case of networks of binary threshold units trained to evaluate Boolean functions. A more general-purpose algorithm is Fahlman and Lebiere's "Cascade-correlation" algorithm (Fahlman & Lebiere, 1990), in which each new unit is trained independently so as to maximize the covariance between its output and the residual error signal. A specific cascaded architecture is generated, in which each newly-recruited unit is connected both to the original inputs and to every pre-existing unit. Hirose et al. (1991) have proposed a comparatively simple variant of back-propagation which allows one to vary the number of units in a single hidden layer during a growing phase, followed by a reduction phase.

In this paper, we describe a new approach based on optimization of an objective function for internal representation, the calculation of which does not

require any explicit calculation of the network's outputs. Because it allows training a network *layer by layer*, this idea lends itself naturally to a constructive strategy; however, mixed strategies, involving e.g., pruning unnecessary units in a layer after the expansion phase, have been considered as well (Section 6).

The concept of direct optimization of internal representations has been previously investigated by several authors, albeit from a different perspective. For example, the CHIR algorithm (Grossman et al. 1989) also treats the internal representations as the basic independent variable of the learning process, but in the context of a fixed architecture, and without explicit identification of an objective function. Krogh et al. (1990) have proposed such a function, which however depends on both the internal representations *and* the connections to the output layer.

3. AN OBJECTIVE FUNCTION FOR INTERNAL REPRESENTATIONS

3.1. Relationships between Discriminant Analysis and Multilayer Classifier Networks

Investigations into the nature of the transformations performed by a multilayer classifier network trained by the least mean squares procedure have revealed interesting connections with what is known in the statistical literature as "discriminant analysis".

Numerical simulations suggest that a network with non-linear hidden units performs more severe feature extraction than linear discriminant analysis, by finding a non-linear transformation which returns a larger value (with no maximization) of a separability criterion involving the scatter matrices of the training vectors (Asoh & Otsu, 1990; Gallinari et al., 1991). It has been shown that, from one layer to the next, internal representations tend to be more and more separated, as clusters become more and more compact, due to the compression effect of the sigmoid function (Gallinari et al., 1991).

Theoretical studies have helped to explain and clarify these experimental results. The linear case was first examined by Gallinari et al. (1988) who proved the formal equivalence between a one hidden layer perceptron and linear discriminant analysis. In this case, it has been shown that minimizing the quadratic error criterion amounts exactly to maximizing the ratio of the between-class to the within-class covariance of the data in the space spanned by the hidden units.

This striking result has been recovered as a special case by Webb and Lowe (1990) who have demonstrated that minimizing the sum-squared error at the output of a network with non-linear hidden nodes is in fact equivalent to maximizing a particular norm, the "network discriminant function", under the condition that the output layer transfer function is linear. In the case of the often used 1 bit encoding, the network discriminant function becomes equal to the trace of the product of the weighted between-class covariance matrix (the weighting being determined by the square of the number of patterns in each class), and the inverse of the total covariance matrix. As a consequence, it was pointed out that, for this particular coding scheme, network classifiers bias strongly in favour of the classes with the largest a priori probabilities.

In this paper, we propose to train each hidden layer of a multilayer perceptron independently, by maximizing the ratio of the between-class to the total covariance in that layer. In the following, the objective function which has been chosen will be presented in greater detail, after which the methodology for optimizing the network structure will be described.

3.2. The Objective Function

Let $\mathbf{Y}$ be a p-dimensional random vector representing the states of the units in a hidden layer, defined as:

$$\mathbf{Y} = f(W \cdot \mathbf{X}) \tag{1}$$

where $\mathbf{X}$ is a random vector containing the outputs of the previous layer, W a weight matrix, and f a transfer function. Following a common convention, the bias terms are considered as weights from a unit that is always on, and are therefore included in W; $\mathbf{X}$ will be referred to as the "input" vector and $\mathbf{Y}$ as its "internal representation".

Assuming the data to be partitioned in N_c classes, we consider the problem of measuring the "separability" of these classes in the space of internal representations. Note that the term "separability" must be understood here as the extent to which the samples from different classes can be separated when projected linearly onto a subspace of the original feature space. It should not be confused with the definition of *linear* separability as the existence of a hyperplane perfectly separating two clusters (see Appendix A).

Noting n_i the a priori probability of class ω_i, the total, between-class and within-class scatter matrices can be defined respectively as:

$$T = E\{\mathbf{Y} - E\{\mathbf{Y}\})(\mathbf{Y} - E\{\mathbf{Y}\})'\} \tag{2}$$

$$B = \sum_{i=1}^{N_c} n_i (E\{\mathbf{Y}|\omega_i\} - E\{\mathbf{Y}\})(E\{\mathbf{Y}|\omega_i\} - E\{\mathbf{Y}\})' \tag{3}$$

$$U = \sum_{i=1}^{N_c} n_i E\{(\mathbf{Y} - E\{\mathbf{Y}|\omega_i\})(\mathbf{Y} - E\{\mathbf{Y}|\omega_i\})'|\omega_i\}, \tag{4}$$

where $E\{\mathbf{Y}\}$ represents the expected value of random vector $\mathbf{Y}$, and $E\{\mathbf{Y}|\omega_i\}$ the expected value of $\mathbf{Y}$ assuming ω_i (as usual, t denotes transposition). It is straightforward to show that $T = U + B$, and that the maximum rank of B is the minimum of $N_c - 1$ and p.

In classical linear discriminant analysis, the problem of finding a discriminant axis is solved by determining a vector $\mathbf{u}$ such that

$$\frac{\mathbf{u}^t B \mathbf{u}}{\mathbf{u}^t U \mathbf{u}}$$

is maximum. Using the fact that $T = U + B$, this is equivalent to maximizing

$$\frac{\mathbf{u}^t B \mathbf{u}}{\mathbf{u}^t T \mathbf{u}}.$$

Noting that $\mathbf{u}$ is defined up to a multiplicative constant, $\mathbf{u}^t B \mathbf{u}$ can be maximized under the constraint $\mathbf{u}^t T \mathbf{u} = 1$. The vector $\mathbf{u}$ is determined accordingly as the eigenvector of matrix $T^{-1}B$ associated with its largest eigenvalue. Consequently, we can define rank(B) discriminant axes associated with the decreasing eigenvalues of $T^{-1}B$. Therefore, linear discriminant analysis can be used as a tool for extracting a limited number of linear features while maximizing the ratio of between-class scatter to within-class scatter.

In the case of two classes, the criterion which is maximized is the eigenvalue of $T^{-1}B$, which is always bounded by 1. When N_c is greater than 2, one has to define a criterion which takes into account the effectiveness of each discriminant axis in showing class separability. In order to formulate such a criterion, we need to convert the scatter matrices into a single number. This number must be larger when the between-class scatter is larger or the within-class scatter smaller. Several criteria have been proposed (Fukunaga, 1972), such as:

$$J_1 = \mathrm{tr}(T^{-1}B)$$
$$J_2 = \mathrm{tr}(B) - \mu(\mathrm{tr}(T) - c)$$
$$J_3 = \frac{\mathrm{tr}(B)}{\mathrm{tr}(T)}$$

where μ is a Lagrange multiplier, and c a constant.

Note that, in the linear case, maximizing J_1 or J_2 can be shown to be equivalent. However, in the non-linear case, the maximization of $\mathrm{tr}(B)$ under the constraint $\mathrm{tr}(T) = c$ is not entirely relevant because there is no longer any reason to fix up the total inertia of the training set after a non-linear transformation. Since on the other hand J_3 depends on the coordinate system while J_1 does not, the latter has been adopted in this study.

4. THE ALGORITHM

Our basic assumption, supported by empirical and theoretical evidence, is that maximizing the separability of representations in a hidden layer renders discrimination easier in the next layer. If enough hidden units are available, or additional hidden layers are added, it is expected that the internal representations will eventually become separable by a single output layer. Since no target value is used, the process of adding hidden layers can be iterated until class labels can be assigned to particular activation patterns, which can be regarded as a kind of self-encoding. If a conventional output coding scheme is preferred, the final representations can optionally be mapped onto imposed target values, using, for example, a standard least mean squares procedure.

Our approach can be described as follows (Figure 1). The initial architecture consists of an input layer and a hidden layer of p_0 units. The weight matrix between this layer and the input layer can be initialized randomly (with, for example, $p_0 = 2$), or can be taken as the optimal transformation matrix resulting from linear discriminant analysis of the input data (in that case, $p_0 = N_c - 1$). A new randomly initialized unit is then added to the layer, and the weights are iteratively updated so as to increase the value of the objective function. If the goal of building a standard multilayer architecture (i.e., without any intra-layer or cross-cut connections) is pursued, each new unit is connected to each unit in the previous layer. Alternatively, a "cascaded" architecture can be obtained by connecting each new unit to the previous layer *and* the existing units in the current layer. Several schemes can be applied for adapting the network after a unit has been added. For example, a computationally effective but very sub-optimal procedure consists of adapting the new unit only, while maintaining the others clamped. It is also possible to apply an alternate direction scheme in which each unit is trained in turn while global optimization is achieved by iterating several times over the set of units. However, we have not found any approach working better on average than the simplest scheme which consists of optimizing the weights in the whole layer simultaneously, starting from the previous configuration. The layer is expanded until the addition of a new unit fails to increase the value of the objective function by a significant amount. The process can then be repeated with a new layer, until again no further improvement in the class separability criterion can be gained. However, it must be noted that a single hidden layer has proved sufficient in all the simulations on the various learning tasks carried out so far. The hidden-to-output weights can be initialized as the eigenvectors of $T^{-1}B$ and refined using, for example, a standard LMS procedure.

```
start
repeat
  add 1 layer
  use output of previous layer as input (add bias term)
  repeat
    add 1 unit in the layer
    optimize the layer
    store weights, value of objective function, and layer output
  until objective function (on learning or cross-validation set) stops increasing
until objective function (on learning or cross-validation set) stops increasing
stop
```

FIGURE 1. The algorithm.

An important remark is that, in real-world problems, the training set is usually contaminated by noise, so that the expansion of the network must be stopped before perfect classification of the training data is achieved. For that reason, it is necessary to test the significance of each new unit by computing the resulting gain in the objective function C *computed on an independent cross-validation set.* Note that this does not imply the calculation of the hidden-to-output weights since the test is made on the objective function and not on the output error.

In order to obtain smaller networks, it may be useful to eliminate the least relevant hidden units after the growing of a layer has been completed. A simple way of pruning the network consists of testing the influence of each hidden unit on the objective function C after a whole layer has been trained. This can be done by removing in turn each unit in the layer and computing the resulting value of C. The unit whose removal causes the least degradation of the objective function is tentatively removed, after which the layer is retrained. If this operation has decreased the value of C by less than some predetermined threshold s, the modification is accepted, and the process is iterated. The modified algorithm with pruning is described in Figure 2.

Different types of search procedures can be used in the optimization phase of the algorithm. Since the analytical expression of the gradient of J_1 with respect to the weights W can be computed (see Appendix B), gradient-based optimization methods can be used, and are probably the most computationally efficient. However, global search algorithms such as simulated annealing or genetic algorithms can be of interest for very hard problems. Good results have been obtained with the evolutionary distributed search algorithm recently proposed by Courrieu (1991). The algorithm has been found to be efficient for hard optimization, and depends only on a very small number of parameters (Figure 3). It has been used successfully in early tests of our method (Lengellé & Denoeux, 1992), but the BFGS second order algorithm has been preferred in subsequent versions.

The complexity of our algorithm scales rather badly with the number p of hidden units, since the computation of the objective function necessitates the inversion of the covariance matrix of size p, which requires $O(p^3)$ operations. For learning tasks requiring large networks, it may therefore be preferable to add new layers rather than to let the number of units in one layer grow indefinitely.

5. EXTENSION TO THE APPROXIMATION OF CONTINUOUS FUNCTIONS

Up to now, we have restricted ourselves to the approximation of decision (Boolean) functions. For this kind of problem, we have proposed to increase, from one layer to the next, a measure of class separability. A similar strategy can readily be proposed for the approximation of any continuous function $F : \mathbb{R}^n \mapsto \mathbb{R}$.

In that case, the goal assigned to each layer can be to increase a measure of linearity between the internal representations and the desired output. A common measure of the linear dependency between a dependent variable F and a set of p controlled variables $Y_1, \ldots, Y_p$ is given by the sample coefficient of multiple determination ρ^2:

$$\rho^2 = 1 - arv \tag{5}$$

with arv, the average relative variance (Weigend et al., 1991), defined by:

$$arv = \frac{\sum_{l=1}^{N} (\hat{F}^{(l)} - F^{(l)})^2}{\sum_{l=1}^{N} (\bar{F} - F^{(l)})^2} \tag{6}$$

```
start
L ← 0
repeat
  add 1 layer ; L ← L + 1
  use output of previous layer as input (add bias term)
  p ← 0
  repeat
    add 1 unit in the layer ; p ← p + 1
    optimize the layer
    store weights, value C of objective function, and layer output
  until objective function stops increasing
  repeat
    for i = 1 to p do
      compute the difference ΔC_i between the objective with and without unit i
    endfor
    select i_min = arg min_{i=1,p} ΔC_i
    optimize the layer without unit i_min
    compute the new value C' of the objective function
    if C − C' < s then
      accept modification ; go-on ← true
    else
      restore previous state ; go-on ← false ; endif
  until go-on
until objective function (on learning or cross-validation set) stops increasing
stop
```

FIGURE 2. The algorithm with pruning.

where $F^{(l)}$ is the value of F for a particular input $X^{(l)}$, $\hat{F}^{(l)}$ is the linear mean square estimate of F given the internal representations $Y_1^{(l)}, \ldots, Y_p^{(l)}$, $\bar{F}$ the average of F on the learning set and N the total number of learning examples.

Therefore, ρ^2 is close to 1 when $\hat{F}^{(l)}$ is a good predictor of $F^{(l)}$, and close to 0 when $\hat{F}^{(l)}$ is a bad predictor. It is interesting to remark (Der Megreditchian, 1983) that ρ^2 can be expressed as:

$$\rho^2 = r^t R^{-1} r \tag{7}$$

with

$$R_{ij} = \frac{\text{cov}(Y_i, Y_j)}{\sqrt{\text{cov}(Y_i, Y_i) \cdot \text{cov}(Y_j, Y_j)}}, \quad i, j = 1, \ldots, p$$

and

$$r_k = \frac{\text{cov}(Y_k, F)}{\sqrt{\text{cov}(Y_k, Y_k) \cdot \text{cov}(F, F)}}, \quad k = 1, \ldots, p.$$

The equivalence between (5) and (7) can easily be demonstrated considering the regression of $\mathbf{F} = (F^{(1)} \ldots F^{(N)})^t$ on the internal representations $\mathbf{Y} = (\mathbf{Y}_1 \ldots \mathbf{Y}_p)$, with $\mathbf{Y}_i = (Y^{(1)} \ldots Y_i^{(N)})^t$. The regression coefficients minimizing $\|\mathbf{F} - \mathbf{Y}A\|^2$ are given by the product of the pseudo-inverse of $\mathbf{Y}$ by $\mathbf{F}$:

$$A = (\mathbf{Y}^t\mathbf{Y})^{-1}\mathbf{Y}^t\mathbf{F} \tag{8}$$

Introducing this expression for A in $\|\mathbf{F} - \mathbf{Y}\mathbf{A}\|^2$ and using the definition of ρ^2 in eqn (5) leads after transformation to eqn (7), which completes the proof.

An immediate consequence of eqns (5) and (6) is that, for a network with one hidden layer and linear outputs, maximizing the sample coefficient of multiple determination between the internal representations and the target values is strictly equivalent to minimizing the sum-squared output error. Consequently, the usual error function of the back-propagation algorithm can be minimized without explicitly introducing the hidden-to-output weights, using eqn (7). The obvious gain is a smaller number of free parameters in the optimization process, from which one can expect a reduction in the number of local minima. However, it must also be noted that this approach requires the inversion of a matrix of

0. **start**
1. Generate a population of K points in weight space according to a preselected "visiting" probability law.
   Calculate the value of the objective function for each of these points and the mean M of those values for the population.
2. **do** K times:
   Take a sample pair of points in the current population by making a random equiprobable choice. Of these two points, select the one that gives the largest value of the objective function, call it the father.
   Replace the other by a realization of a multidimensional Gaussian variable with mean equal to the father and variance $\sigma^2 I$.
   Calculate the objective function for the new point and update M.
3. **if** M has improved, **goto** 2.
4. Decrease σ.
5. **if** the end of search criterion is verified, **goto** 7.
6. **goto** 2
7. **stop**

FIGURE 3. The evolutionary distributed search algorithm

size p at each evaluation of the objective function or its gradient, which is given in Appendix B.

These considerations can easily be extended to the approximation of a function $G: \mathbb{R}^n \mapsto \mathbb{R}^m$. In this case, the global sum-squared error is the sum over all output neurons of the individual sum-squared errors. If the function G has the same dispersion over all its components (which can always be achieved by normalization), maximizing the sum of the ρ_i^2 between the internal representations and the ith component of G is equivalent to minimizing the global sum-squared output error.

As a consequence, the same strategy as described in the previous section can be used for training a network and determining its structure layer by layer, when approximating a continuous function. Once the objective function cannot be increased any longer from one layer to the next, the last layer of weights can be computed by solving a least squares problem between the vector of states in the last layer and the target output. With linear output units the solution is obtained in one step using a pseudo-inverse approach.

6. EXPERIMENTS

The various issues discussed in this paper will now be illustrated using a series of numerical experiments. The first learning task that will be considered is the so-called "two spirals" problem, which consists in separating two interlocking spirals as shown in Figure 4. The complexity of this learning task is such that the standard back-propagation algorithm fails to solve it using as many as 50 randomly initialized hidden units (Baum & Lang, 1991). Successful results have been reported with the cascade-correlation algorithm (Fahlman & Lebiere, 1990) which generates for this problem between 12 and 19 (mean: 15.2) partially interconnected hidden units corresponding to at least 114 connection weights.

Results with our method are displayed in Figure 5. An interesting fact about these results is the remarkably small variability of the learning curves from one run of the algorithm to the other. With 33 hidden units, all the networks generated correctly classified each of the 192 data points (see one solution in Figures 4 and 6).

A comparison with cascade-correlation (CC) has been performed on this learning task (Table 1). Each algorithm was run 10 times. The statistics reported in Table 1 were computed using the successful trials only, i.e., eight trials for CC and the 10 trials for our algorithm (runs of the CC algorithm which had not converged with 30 hidden units have been classified as unsuccessful). The simulations were performed in the MATLAB language on a Sun workstation. The execution times given in Table 1 must be interpreted with great caution, as the program codes have not been optimized with respect to execution speed. As can be seen, our algorithm most of the time generated smaller networks, in terms of number of connections. However, it was about three times slower (in our implementation), mainly because of the matrix inversion needed for each computation of the objective function.

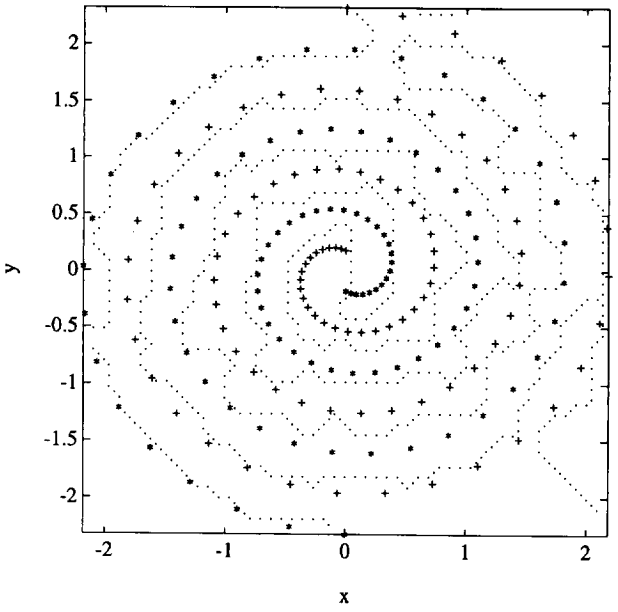

FIGURE 4. The spirals data, with one of the decision boundaries obtained (class 1: +; class 2:•).

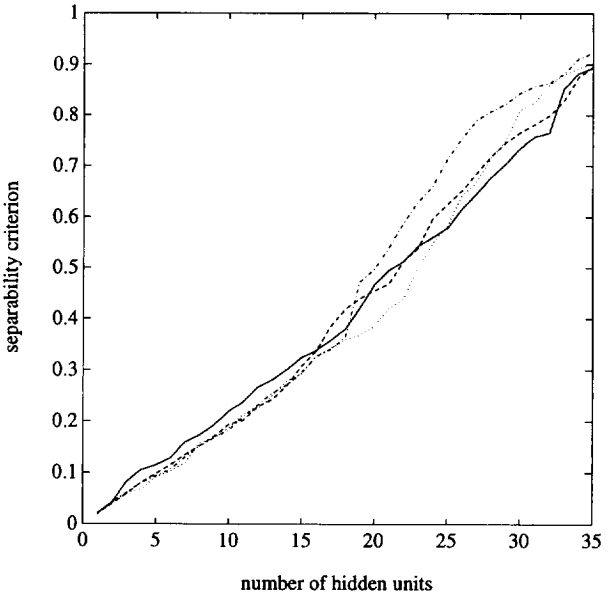

FIGURE 5. $Tr(T^{-1}B)$ as a function of the number of hidden units for the spirals problem (four consecutive runs).

The ability of our algorithm to discover interesting solutions to discrimination problems using a comparatively small number of hidden units has been confirmed by several other experiments. For example, Figure 7 represents a two-class problem in which the data points are arranged along two sine curves. At least two obvious solutions to this problem, which are easily found by the back-propagation algorithm, make use of five hidden units. Shown in Figures 7 (hidden layer weights and decision boundary) and 8 (network output) is a rather unexpected solution with only three hidden units which is discovered by our algorithm most of the time, starting from random initial conditions. After running the back propagation algorithm many times, no comparable solution has been obtained. Among other things, this example demonstrates that the common interpretation of hidden units as performing piecewise linear discrimination is not always relevant.

A totally different example is taken from Kohonen et al. (1989). In this problem, the samples to be classified are assumed to be taken from two symmetrical bidimensional normal distributions having the same mean and square root of variance equal to 1 and 2, respectively (Figure 9). In this case, the optimal decision boundary corresponding to the Bayes classifier is a circle. Therefore, the complexity of the task arises this time not so much from the complexity of the decision boundary, but from the very strong overlap between the two distributions. In such a situation, it is necessary to use a resampling or cross-validation method to estimate the class separability criterion to avoid the over-learning effect. Figure 10 shows the means for 10 trials of the estimated criteria on the training set and two independent data sets (which can be considered as a cross-validation set and a test set), as a function of the number of hidden units. Figure 11 reports the corresponding estimated error rates. As can be seen a network with 3–4 hidden units would be selected by this method, yielding an estimated error rate of approximately 0.29, i.e., slightly more than the Bayes error rate (0.26).

As an example of a complex real-world classification problem, the application to automatic analysis of human sleep was considered. Sleep research has considerably improved in the last 10 years. In humans, polygraphic sleep techniques have permitted the definition of five sleep stages, described

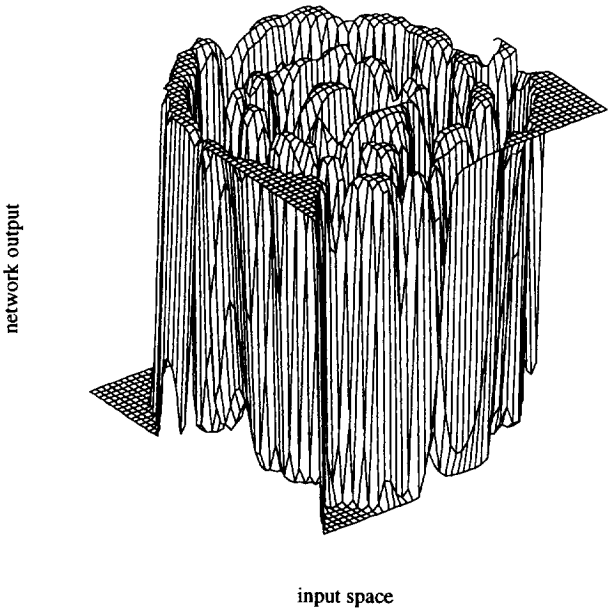

FIGURE 6. Output of a network trained on the spirals data.

TABLE 1
Comparison between Cascade-correlation (CC) and our Algorithm (IRO) on the Spirals Problem

	Hidden Units	Connections	Execution Time ($\times 10^4$ s)
CC			
Min.	13	133	0.14
Max.	20	273	3.77
Mean	15.63	181.75	1.00
Std	2.13	23.90	1.15
IRO			
Min.	30	121	1.65
Max.	42	169	4.34
Mean	36.60	147.40	2.93
Std	4.03	16.13	0.84

The statistics have been computed using 8 trials for CC and 10 trials for IRO

in a standard manual (Rechstschaffen & Kales, 1968). Each sleep stage is usually hand scored by an expert using 30-s epochs, for the whole night. Representation of sleep stage versus time is called a hypnogram. A neural network approach to automatic analysis of sleep has been recently proposed (Schaltenbrand et al., 1993). In this study, 17 parameters were extracted from the time-frequency analysis of three electro-physiological signals: electro-encephalogram (EEG), electro-myogram (EMG) and electro-oculogram (EOG), and were used as inputs to a multilayer perceptron. The architecture of the network, optimized using a cross-validation data set, consisted in a single hidden layer 17-10-6 network (the output layer was

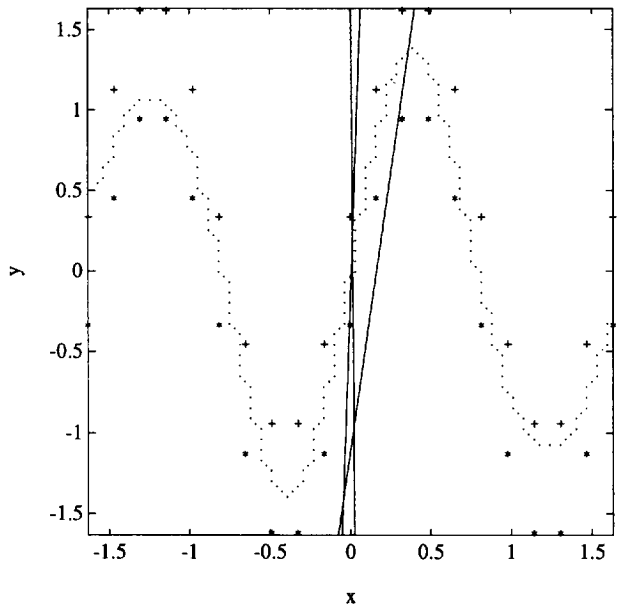

FIGURE 7. The two sine curves data, with a particular solution (weights and corresponding decision boundary) involving only three hidden neurons (class 1: +; class 2: *).

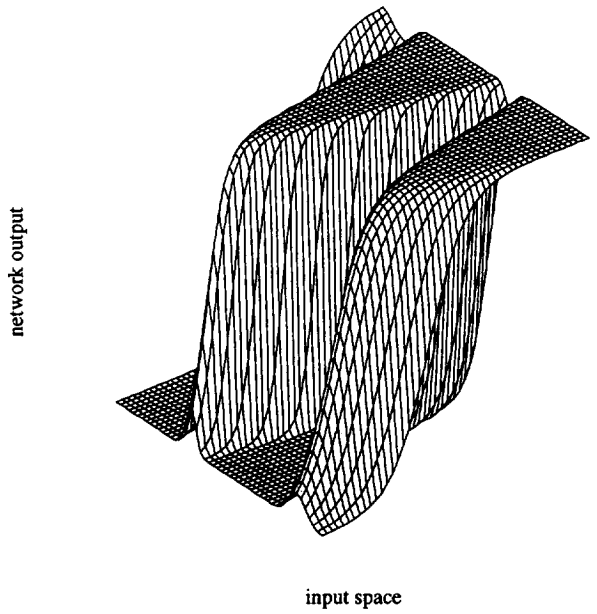

FIGURE 8. Output of a network trained on the two sine curves data.

composed of six units corresponding to five sleep stages and wakeness). The network was trained on a data set composed of 12,455 30-s epochs corresponding to the analysis of 12 all-night sleep recordings, and tested on an independent test set of 11,906 30-s epochs. Each night of the training set was analysed by a single expert. The test set was labelled by a pool of 10 experts and observa-

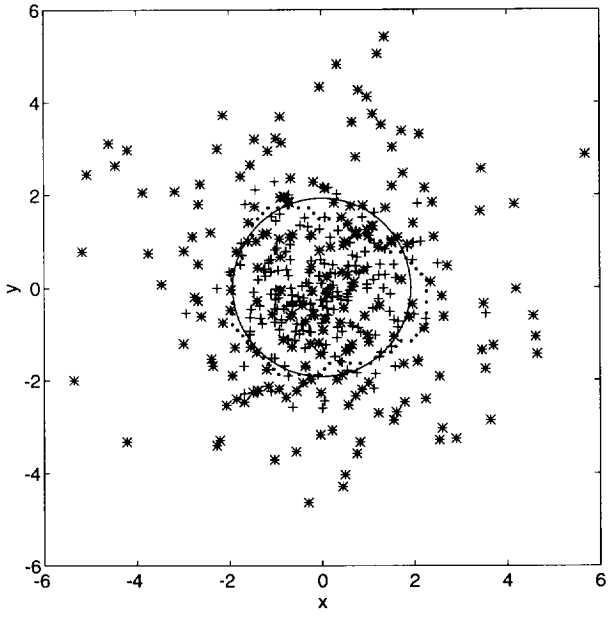

FIGURE 9. The data set for the third problem: two symmetrical bidimensional normal distributions with the same mean and square root of variance equal to 1 and 2, respectively (class 1: +; class 2: *). The Bayes boundary and the boundary obtained are indicated by solid and dotted lines, respectively.

tions were assigned class labels according to the majority rule (notice that the average inter-expert agreement was shown to be 87.9% in this study). The authors have reported an average misclassification rate of 14.9% on the training set and 19.4% on the test set.

We have applied our internal representation optimization algorithm to this problem, using the same data sets. The network architecture was optimized accordingly, consisting of a single hidden layer of 12 units, i.e., slightly more than the network reported by Schaltenbrand. We noticed a training set error rate of 19.6% (to compare with 14.9%) and a test set error rate of only 17.1%, i.e., smaller than the error probability reported by Schaltenbrand (19.4%), and also smaller than the error rate obtained on the training set. This is certainly due to a regularization effect of the labelling process on the solution, but however indicates that our algorithm seems to be less sensitive to overfitting.

As a case study for the approximation of a continuous function, let us now consider the function:

$$F(x, y) = \sin(\sqrt{x^2 + y^2})$$

in the domain $[-5, 5]^2$ (Figure 12).

The evolution of ρ^2 as a function of the number of hidden units when training five networks on the function F starting from different initial conditions is reported in Figure 13. Once again,

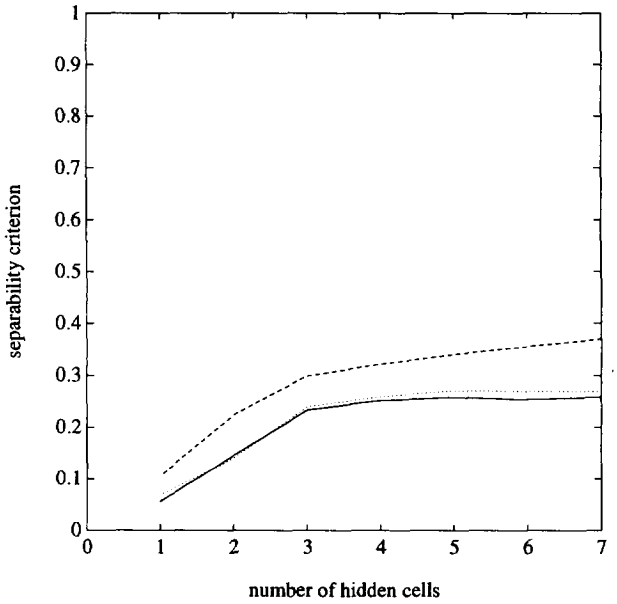

FIGURE 10. $\mathrm{Tr}(T^{-1}B)$ as a function of the number of hidden units for the third problem (average over 10 trials), on the training set (upper curve) and two independent sets (lower curves).

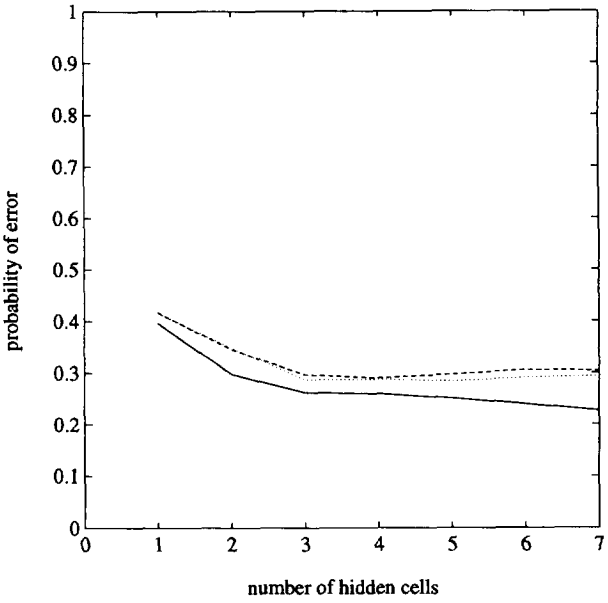

FIGURE 11. Misclassification error rate on the learning set (lower curve) and two independent test sets (upper curves), as a function of the number of hidden units, for the third problem (average over 10 trials).

robustness appears to be a noticeable feature of the algorithm. A sample coefficient of multiple determination of 0.99 is always attained with 21 hidden units. In order to evaluate the influence of the objective function on the efficiency of the algorithm, the same constructive procedure has been applied while minimizing the mean-squared output error *mse* instead of maximizing ρ^2. Since it has been shown in Section 5 that the maximization of ρ^2 and the minimization of *mse* are equivalent, the only difference lies in the number of free parameters of the optimization problem. As expected, the performance of the algorithm was significantly worse when *mse* was taken as the objective function, with a maximum value of ρ^2 equal to 0.95 with 21 hidden units. This confirms that the choice of ρ^2 as the objective function is to some extent responsible for the robustness of the algorithm.

Figure 14 shows the evolution of ρ^2 during the growing phase followed by a reduction phase conducted according to the pruning procedure described in Section 4 (with the stopping criterion relaxed). In the case, the value of 0.99% for ρ^2 had been reached with 21 hidden units in the growing phase, and was preserved with only 15 hidden units in the decreasing phase. The value of 0.9975 obtained with 25 hidden units in the increasing phase could be maintained after the pruning of 3 hidden units. These results demonstrate the usefulness of a reduction phase for obtaining minimal-size networks.

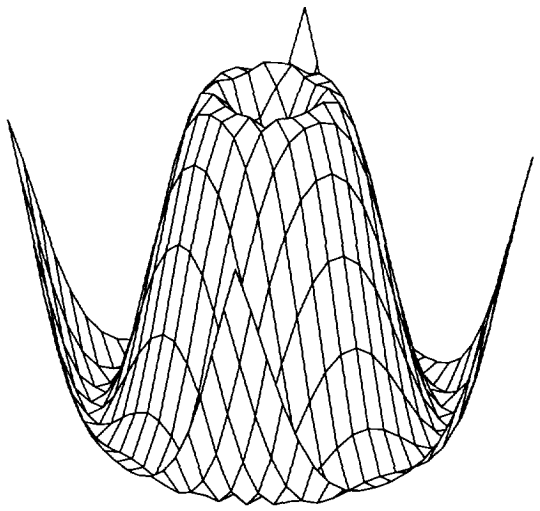

FIGURE 12. The function $F(x, y) = \sin((x^2 + y^2)^{1/2})$ in the domain $[-5, 5]^2$.

7. CONCLUSIONS

A new algorithm for adapting the topology of multilayer feedforward networks in the course of the training process has been described. The basic features of this algorithm are an objective function for internal representations, and a scheme for gradually increasing the complexity of the network architecture. Objective functions have been defined both for classification and approximate interpolation problems. These functions, which do not depend on the hidden-to-output weights, can be regarded as intrinsic measures of the degree to which the task can be solved linearly in the space spanned by the hidden units. The process of adding new units—and, if necessary, new layers—stops when the last hidden layer can be connected directly to the output layer.

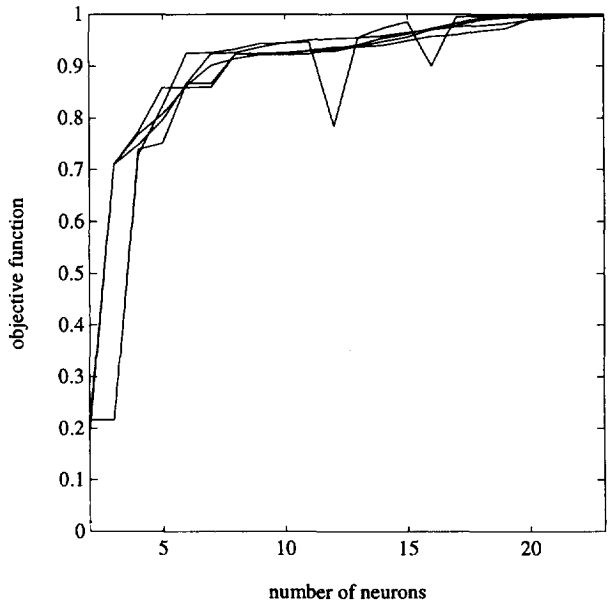

FIGURE 13. ρ^2 as a function of the number of hidden units (five consecutive runs).

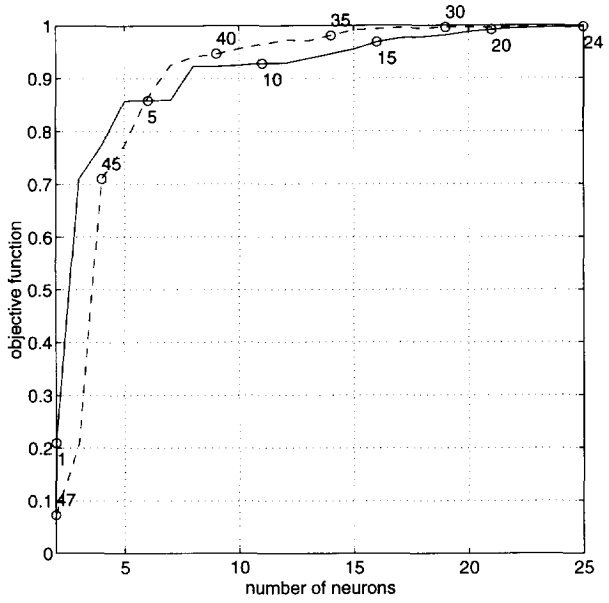

FIGURE 14. Evolution of ρ^2 during a growing phase (solid line) followed by a pruning phase (dotted line). The numbers indicate the iterations of the construction algorithm. Iterations from 1 to 24 correspond to the growing phase. Iterations from 25 to 47 correspond to the pruning phase.

Many variants of this basic scheme can be imagined, only a few of which have been fully investigated. One of these variants, which has proved beneficial, consists of reducing the size of the hidden layers generated by the constructive algorithm, by gradually removing the least relevant units and retraining the whole layer. Another interesting possibility could be to extend the search space of the algorithm to different topologies (e.g., short-cut connections, cascaded architectures) and different types of units (e.g., radial basis function units). This possibility—and others—will be subject to further research.

REFERENCES

Asoh, H., & Otsu, N. (1990). An approximation of nonlinear discriminant analysis by multilayer neural networks. In *Proceedings IJCNN'90*, San Diego (pp. III-211–216).

Baum, E. B., & Haussler, D (1989). What size net gives valid generalization. *Neural Computing*, 1(1), 151–160.

Baum, E. B., & Lang, K. J. (1991). Constructing hidden units using examples and queries. In R. P. Lippman, J. E. Moody, & D. S. Touretzky (Eds.), *Neural information processing 3* (pp. 904–910). San Mateo, CA: Morgan Kaufmann.

Burton, R. M., & Farris, W. G. (1991). Reliable evaluation of neural networks. *Neural Networks*, 4, 411–415.

Courrieu, P. (1991). A distributed search algorithm for hard

optimization. Technical Report TA9101, CREPCO, Université de Provence, F-13621 Aix-en-Provence Cedex 1.

Der Megreditchian, G (1983). Une identité algébrique intéressante pour la statistique mathématique. *La Météorologie*, VI(32), 5–14.

Duda, R. O., & Hart, P. E. (1973). *Pattern classification and scene analysis*. New York: John Wiley.

Fahlman, S. E., & Lebiere, C. (1990). The cascade-correlation learning architecture. In D. S. Touretzky (Ed.), *Advances in neural information processing systems 2* (pp. 524–532). San Mateo, CA: Morgan Kaufmann.

Fukunaga, K. (1972). *Introduction to statistical pattern recognition*. Electrical Science. New York: Academic Press.

Gallinari, P., Thiria, S., & Fogelman-Soulie, F. (1988). Multilayer perceptrons and data analysis. In *Proc. ICNN'88*, San Diego (pp. 391–398).

Gallinari, P., Thiria, S., Badran, F., & Fogelman-Soulie, F. (1991). On the relations between discriminant analysis and multilayer perceptrons. *Neural Networks*, **4**, 349–360.

Grossman, T., Meir, R., & Domany, E. (1989). Learning by choice of internal representations. In D. S. Touretzky, (Ed.), *Advances in neural information processing systems 1* (pp. 73–80). San Mateo, CA: Morgan Kaufmann.

Haussler, D. (1988). Quantifying inductive bias: AI learning algorithms and valiant's learning framework. *Artificial Intelligence*, **36**, 177–221.

Hinton, G. E. (1990). Connectionist learning procedures. In Y. Kodratoff & R. Michalski (Eds), *Machine learning III*. (pp. 555–610). San Mateo, CA: Morgan Kaufmann.

Hirose, Y., Yamashita, K., Hijiya, S. (1991). Back-propagation algorithm which varies the number of hidden units. *Neural Networks*, **4**(1), 61–66.

Hornik, K. (1991). Approximation capabilities of multilayer feedforward networks. *Neural Networks*, **4**, 251–257.

Kohonen, T., Chrisley, R., & Barna, G. (1989). Statistical pattern recognition with neural networks: Benchmarking studies. In L. Personnaz & G. Dreyfus (Eds.), *Neural networks from models to applications*. Paris: I.D.S.T.

Krogh, A., Thorebegsson, G. I., & Hertz, J. A. (1990). A cost function for internal representations. In D. S. Touretzky (Ed.) *Advances in neural information processing systems 2* (pp. 733–740). San Mateo, CA: Morgan Kaufmann.

Lengellé, R., & Denoeux, T (1992). Optimizing multilayer networks layer per layer without back-propagation. In Igor Aleksander & John Taylor (Eds.), *Artificial neural networks II* (pp. 995–998). Amsterdam: North-Holland.

Marchand, M., Golea, M., & Ruján, P. (1990). A convergence theorem for sequential learning in two-layer perceptrons. *Europhysics Letters*, **11**(6), 487–492.

Mézard, M., & Nadal, J. P. (1989). Learning in feedforward layered networks: The tiling algorithm. *Journal of Physics A: Mathematical and General*, **22**, 2191–2203.

Rechstschaffen, A. & Kales, A. (1968) *A manual of standardized terminology, techniques and scoring system for sleep stages of human subjects*. Public Health Service, Washington, DC: U.S. Government Printing Office.

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, **65**, 386–408.

Rumelhart, D. E. Hinton, G. E., & Williams, R. J. (1986). Learning internal representations by error propagation. In D. E. Rumelhart & J. McClelland (Eds), *Parallel distributed processing*. Cambridge, MA: MIT press.

Schaltenbrand, N., Lengellé, R., & Macher, J. P. (1993). Neural network model: Application to automatic analysis of human sleep. *Computers and Biomedical Research*, **26**, 157–171.

Sontag, E. D. (1991). Remarks on interpolation and recognition using neural nets. In R. L. Lippman, J. E. Moody, & D. S. Touretzky (Eds.), *Advances in neural information processing systems 3* (pp. 939–945). San Mateo, CA: Morgan Kaufmann.

Vapnik, V. N. (1983) *Estimation of dependences based on empirical data*. Springer series in statistics. Berlin: Springer-Verlag.

Webb, A. R., & Lowe, D. (1990). The optimised internal representation of multilayer classifier networks performs non linear discriminant analysis. *Neural Networks*, **3**, 367–375.

Weigend, A.S., Rumelhart, D.E., & Huberman, B.A. (1991). Generalization by weight-elimination with application to forecasting. In R. P. Lippman, J. E. Moody, & D. S. Touretzky (Eds), *Neural Information Processing*, **3**, (pp. 875–882). San Mateo, CA: Morgan Kaufmann.

Zollner, R., Schmitz, H. J., Wünsch, F., & Krey, U. (1992). Fast generating algorithm for a general three-layer perceptron. *Neural Networks*, **5**, 771–777.

NOMENCLATURE

$\mathbf{Y}$	vector of unit states in the current hidden layer
$\mathbf{X}$	outputs from previous layer
p	number of units in the current hidden layer
W	weight matrix
f	hidden unit transfer function
N_c	number of classes
T	total scatter matrix
B	between-class scatter matrix
U	within-class scatter matrix
$J_i (i = 1,2,3)$	separability criteria
N	number of training samples
ρ^2	sample coefficient of multiple determination
arv	average relative variance
R	correlation matrix of vector $\mathbf{Y}$

APPENDIX A: RELATIONSHIP BETWEEN LINEAR DISCRIMINATION AND FISHER'S LINEAR DISCRIMINANT

Consider the case where two clusters are to be linearly separated. Let X_i be the vector composed of the p components of the observation i and 1 as the $(p+1)$th component (this transformation allows one to consider only hyperplanes passing through the origin in the new space even though the corresponding hyperplane can be in any position in the initial space). The linear discrimination problem consists in determining a hyperplane defined by its normal vector $\mathbf{A}$ such that $\mathbf{X}_i^t\mathbf{A} > 0$ if $\mathbf{X}_i$ belongs to class one and $\mathbf{X}_i^t\mathbf{A} < 0$ if the observation belongs to class 2. Replacing $\mathbf{X}_i$ by $-\mathbf{X}_i$ if $\mathbf{X}_i$ belongs to cluster 2 transforms the initial problem into a new one that consists of finding now a vector $\mathbf{A}$ verifying $\mathbf{X}_i^t\mathbf{A} > 0$ for all i. Solutions, if they exist (i.e., if and only if the clusters are linearly separable), can be found by solving this system of inequalities. An approach to determine a solution is the perceptron learning rule, which consists of modifying $\mathbf{A}$ in such a way that each misclassified point attempts to verify its own inequality. This procedure can be shown to be equivalent to minimizing the sum of the distances from these points to the decision boundary. However, since no weight vector can correctly classify every sample in a non-separable set, the corrections may never cease. This is the reason why researchers have tried to define criteria that take into account all the samples instead of focusing their attention only on misclassified samples.

Let us attempt to solve $\mathbf{X}_i^t \mathbf{A} = b_i$, where b_i is an arbitrarily positive number. Generally, the number N of samples is much larger than the dimension $p+1$ of $\mathbf{A}$, so that no exact solution exists. However, we can seek a vector that attempts to minimize the sum of squares between $\mathbf{X}_i^t \mathbf{A}$ and b_i. The solution is then given by a pseudo-inverse approach. An essential result (Duda & Hart, 1973) is that if $b_i = 1/N_1$ if the observation X_i belongs to class 1 and $1/N_2$ if it belongs to class 2, where N_j is the number of observations from class j, then the weight vector minimizing the error is proportional to the product of the inverse of the within-class scatter matrix by the difference of the mean vectors $[\mathbf{A} = U^{-1}(m_1 - m_2)]$ which is precisely the solution for Fisher's linear discriminant (Duda & Hart, 1973). This relationship justifies the definition of some criterion involving scatter matrices as a measure of class separability.

APPENDIX B: GRADIENT CALCULATION

Determination of the Gradient of $\mathrm{tr}(T^{-1}B)$ w.r.t. the Weights

Let $\mathbf{y}$ and $\mathbf{Y}$ denote respectively the p-dimensional vectors of activations and states of a hidden layer, and $\mathbf{X}$ the vector of inputs to that layer. By definition, we have:

$$\mathbf{y} = W \cdot \mathbf{X}, \tag{B.1}$$

$$\mathbf{Y} = f(\mathbf{y}), \tag{B.2}$$

where W is the weight matrix, and f the transfer function.

Assuming the input $\mathbf{X}$ to be taken from a given distribution, let us consider the total scatter matrix T and the between-class scatter matrix B in the hidden layer, as well as $\mathrm{tr}(T^{-1}B)$, as functions of the weights W, and let $g(W) = \mathrm{tr}(T^{-1}B)$. Using the fact that both T and B are symmetric, the derivative of g w.r.t. a weight $W_{l,k}$ is given by:

$$\frac{\partial g}{\partial W_{l,k}} = \sum_{i=1}^{p} \sum_{j=1}^{i} \frac{\partial_g}{\partial T_{i,j}} \frac{\partial T_{i,j}}{\partial W_{l,k}} + \sum_{m=1}^{p} \sum_{n=1}^{m} \frac{\partial g}{\partial B_{m,n}} \frac{\partial B_{m,n}}{\partial W_{l,k}}. \tag{B.3}$$

Let us now express the four derivatives appearing in (B.3). We use the fact that, for any non-singular matrix R:

$$\frac{\partial R^{-1}}{\partial x} = -R^{-1} \frac{\partial R}{\partial x} R^{-1}.$$

Applying this equation for $x = R_{i,j}$ yields:

$$\frac{\partial R^{-1}}{\partial R_{i,j}} = -R^{-1} I_{i,j} R^{-1},$$

where $I_{i,j}$ is the matrix with the component (i,j) equal to 1, and all other components equal to 0.

Furthermore, if R is symmetric, then:

$$\frac{\partial R}{\partial R_{i,j}} = I_{i,j} + I_{j,i} - \delta_{i,j} I_{i,j} = I^*_{i,j},$$

where $\delta_{i,j}$ is the Kronecker symbol, and consequently:

$$\frac{\partial R^{-1}}{\partial R_{i,j}} = -R^{-1} I^*_{i,j} R^{-1}.$$

From the preceding considerations, we have:

$$\frac{\partial g}{\partial T_{i,j}} = -\mathrm{tr}(T^{-1} I^*_{i,j} T^{-1} B) \tag{B.4}$$

$$= -\mathrm{tr}(I^*_{i,j} T^{-1} B T^{-1}) \tag{B.5}$$

$$= -(\theta_{i,j} + \theta_{j,i} - \delta_{i,j}\theta_{i,j}) \tag{B.6}$$

$$= -(2 - \delta_{i,j})\theta_{i,j}, \tag{B.7}$$

where θ_{ij} is the element (i,j) of matrix $\theta = T^{-1}BT^{-1}$. Moreover,

$$\frac{\partial T_{i,j}}{\partial W_{l,k}} = \frac{\partial \mathrm{cov}(Y_i, Y_j)}{\partial W_{l,k}} \tag{B.8}$$

$$= \mathrm{cov}(Y_i, f'(y_l) X_k)\delta_{l,j} + \mathrm{cov}(f'(y_l) X_k, Y_j)\delta_{l,i} \tag{B.9}$$

where $\mathrm{cov}(\cdot, \cdot)$ is the usual covariance operator.

From the properties recalled above:

$$\frac{\partial g}{\partial B_{m,n}} = T^{-1}_{m,n} + T^{-1}_{n,m} - \delta_{m,n} T^{-1}_{m,n} \tag{B.10}$$

$$= (2 - \delta_{m,n}) T^{-1}_{m,n}. \tag{B.11}$$

Let us now give a formula for

$$\frac{\partial B_{m,n}}{\partial W_{l,k}}.$$

By definition, we have:

$$B = \sum_{i=1}^{N_c} n_i (E\{\mathbf{Y}|\omega_i\} - E\{\mathbf{Y}\})(E\{\mathbf{Y}|\omega_i\} - E\{\mathbf{Y}\})^t.$$

Hence,

$$\frac{\partial B_{m,n}}{\partial W_{l,k}} = \sum_i n_i \left(\left(\frac{\partial (E\{\mathbf{Y}|\omega_i\} - E\{\mathbf{Y}\})}{\partial W_{i,k}} \right)_m (E\{\mathbf{Y}|\omega_i\} - E\{\mathbf{Y}\})_n \right) + \sum_i n_i \left((E\{\mathbf{Y}|\omega_i\} - E\{\mathbf{Y}\})_m \left(\frac{\partial (E\{\mathbf{Y}|\omega_i\} - E\{\mathbf{Y}\})}{\partial W_{i,k}} \right)_n \right)$$

with

$$\frac{\partial E\{\mathbf{Y}|\omega_i\}_m}{\partial W_{l,k}} = E\{f'(y_m)\mathbf{X}_k|\omega_i\}\delta_{l,m}$$

and

$$\frac{\partial E\{\mathbf{Y}\}_m}{\partial W_{l,k}} = \frac{\sum_i n_i \frac{\partial E\{\mathbf{Y}|\omega_i\}_m}{\partial W_{l,k}}}{\sum_i n_i}$$

which completes the calculation of

$$\frac{\partial g}{\partial W_{l,k}}.$$

The complexity of this gradient calculation can be estimated in the following way. Let N be the size of the training set and N_c the number of classes. The estimation of T requires

$$N \frac{p(p+1)}{2}$$

multiplications, and the determination of B about

$$N_c \frac{p(p+1)}{2}.$$

The objective function can be evaluated with $O(p^3)$ more operations (inversion of T and product with B). The first step for gradient determination is the evaluation of matrix θ, with $2p^3$ multiplications. Then, for each weight, the main part of the computational time is spent in the evaluation of the covariances appearing in eqn (B.9) ($3N$ operations) and of

$$\frac{\partial E\{Y|\omega_i\}_m}{\partial W_{l,k}},$$

i.e., N products in the case of equal priors. This must be multiplied by $o(p^2)$ because of the double summations in eqn (B.3). Consequently, the computation of the whole gradient requires $O(Np^3)$ operations.

Determination of the Gradient of ρ^2 w.r.t. the Weights

With the notations introduced in Section 5, we have:

$$\rho^2 = r^t R^{-1} r$$

with:

$$R_{ij} = \frac{\text{cov}(\mathbf{Y}_i, \mathbf{Y}_i)}{\sqrt{\text{cov}(\mathbf{Y}_i, \mathbf{Y}_i) \cdot \text{cov}(\mathbf{Y}_j, \mathbf{Y}_j)}},\ i,j = 1,\ldots,p \quad \text{and}$$
$$r_k = \frac{\text{cov}(\mathbf{Y}_k, F)}{\sqrt{\text{cov}(\mathbf{Y}_k, \mathbf{Y}_k)\text{cov}(F, F)}},\ k = 1,\ldots,p.$$

In order to compute the gradient of ρ^2 w.r.t. the weights W, we shall use the following property (Der Megreditchian, 1983): Let $\bar{R}^{(l)}$ be the matrix obtained by removing from R line l and column l, $\bar{r}^{(l)}$ the vector obtained by removing from r the component l, and $s^{(l)}$ the vector formed by the lth column of $\bar{R}^{(l)}$. We have:

$$\rho^2 = \bar{r}^{(l)t} \bar{R}^{(l)-1} \bar{r}^{(l)} + \frac{(r_l - s^{(l)t} \bar{R}^{(l)-1} \bar{r}^{(l)})^2}{1 - s^{(l)t} \bar{R}^{(l)-1} s^{(l)}}$$

which can be more simply written:

$$\rho^2 = \bar{\rho}^2 + \frac{(r_l - A)^2}{1 - B}.$$

In this expression, only r_l, A and B depend on the weights $W_{l,1},\ldots,W_{l,n}$ connecting the lth hidden unit to the previous layer. Therefore, we have:

$$\frac{\partial \rho^2}{\partial W_{l,p}} = \frac{2(R_l - A)\left(\frac{\partial r_l}{\partial W_{l,p}} - \frac{\partial A}{\partial W_{l,p}}\right)(1-B) + (r_l - A)^2 \frac{\partial B}{\partial W_{l,p}}}{(1-B)^2}.$$

We must now determine

$$\frac{\partial r_l}{\partial W_{l,p}},\ \frac{\partial A}{\partial W_{l,p}}$$

and

$$\frac{\partial B}{\partial W_{l,p}}.$$

We have:

$$\frac{\partial r_l}{\partial W_{l,p}} = \frac{\text{cov}(F, f'(y_l) X_p)\sqrt{\text{cov}(Y_l, Y_l)} - \text{cov}(Y_l, F) - \text{cov}(Y_l, f'(y_l) X_p)/\sqrt{\text{cov}(Y_l, Y_l)}}{\text{cov}(Y_l, Y_l)\sqrt{\text{cov}(F, F)}}$$

and

$$\frac{\partial A}{\partial W_{l,p}} = \frac{\partial}{\partial W_{l,p}}(s^{(l)} \bar{R}^{(l)-1} \bar{r}^{(l)t}) = (\nabla W_{l,p} s^{(l)t}) \bar{R}^{(l)-1} \bar{r}^{(l)t}$$

with

$$\frac{\partial s_k^{(l)}}{\partial W_{l,p}} = \frac{\partial}{\partial W_{l,p}}\left(\frac{\text{cov}(Y_1, Y_k)}{\sqrt{\text{cov}(Y_k, Y_k) \cdot \text{cov}(Y_1, Y_1)}}\right).$$

Denoting by *Num* and *Den*, respectively, the numerator and denominator of the previous expression, we can write:

$$\frac{\partial s_k^{(l)}}{\partial W_{l,p}} = \frac{\frac{\partial Num}{W_{l,p}} Den - Num \frac{\partial Den}{W_{l,p}}}{Den^2}$$

where

$$\frac{\partial Num}{\partial W_{l,p}} = \text{cov}(Y_k, f'(y_l) X_p)$$

and

$$\frac{\partial Den}{\partial W_{l,p}} = \frac{\text{cov}(Y_k, Y_k)\,\text{cov}(Y_l, f'(y_l) X_p)}{Den}.$$

Lastly, for

$$\frac{\partial B}{\partial W_{l,p}},$$

we simply obtain:

$$\frac{\partial B}{\partial W_{l,p}} = 2 s^{(l)} \bar{R}^{(l)-1} \nabla W_{l,p} s^{(l)t}$$

The computation of the objective function requires

$$N \frac{p(p+1)}{2}$$

multiplications for the determination of R, $O(p^3)$ for the matrix inversion and $O(p^2)$ for the evaluation of $r^t R^{-1} r$, where N is the number of samples used to train the network. To evaluate the gradient, all the covariances appearing in the equations

can be determined with $O(Np^2)$ operations and all the $\bar{R}^{(l)-1}$ must be computed with (at most) $O(p^4)$ multiplications. Computation of

$$\frac{\partial A}{\partial W_{i,p}}$$

and

$$\frac{\partial B}{\partial W_{i,p}}$$

need $O(p^2)$ operations. Consequently, the whole gradient determination can be achieved with $O(p^4 + Np^2)$ multiplications.